Optimal-depth sorting networks

Bundala, Daniel; Codish, Michael; Cruz-Filipe, Luís; Schneider-Kamp, Peter; Závodný, Jakub

Published in:
Journal of Computer and System Sciences

DOI:
10.1016/j.jcss.2016.09.004

Publication date:
2017

Document version
Early version, also known as pre-print

Citation for pulished version (APA):
Bundala, D., Codish, M., Cruz-Filipe, L., Schneider-Kamp, P., & Závodný, J. (2017). Optimal-depth sorting networks. Journal of Computer and System Sciences, 84, 185-204. DOI: 10.1016/j.jcss.2016.09.004

Optimal-Depth Sorting Networks*

Daniel Bundala
Department of Computer Science
University of Oxford

Michael Codish
Department of Computer Science
Ben-Gurion University of the Negev

Luís Cruz-Filipe
Dept. Mathematics and Computer Science
University of Southern Denmark

Peter Schneider-Kamp
Dept. Mathematics and Computer Science
University of Southern Denmark

Jakub Závodný
Department of Computer Science
University of Oxford

Abstract

We solve a 40-year-old open problem on the depth optimality of sorting networks. In 1973, Donald E. Knuth detailed, in Volume 3 of "*The Art of Computer Programming*", sorting networks of the smallest depth known at the time for $n \leq 16$ inputs, quoting optimality for $n \leq 8$. In 1989, Parberry proved the optimality of the networks with $9 \leq n \leq 10$ inputs. In this article, we present a general technique for obtaining such optimality results, and use it to prove the optimality of the remaining open cases of $11 \leq n \leq 16$ inputs. We show how to exploit symmetry to construct a small set of two-layer networks on n inputs such that if there is a sorting network on n inputs of a given depth, then there is one whose first layers are in this set. For each network in the resulting set, we construct a propositional formula whose satisfiability is necessary for the existence of a sorting network of a given depth. Using an off-the-shelf SAT solver we show that the sorting networks listed by Knuth are optimal. For $n \leq 10$ inputs, our algorithm is orders of magnitude faster than the prior ones.

1 Introduction

General-purpose sorting algorithms are based on comparing and exchanging pairs of inputs. If the order of these comparisons is predetermined by the number of inputs to sort and does not depend on their concrete values, then the algorithm is said to be data oblivious. Such algorithms are well-suited for e.g. parallel sorting or secure multi-party computations, unlike standard sorting algorithms, such as QuickSort, MergeSort or HeapSort, where the order of comparisons performed depends on the input data.

Sorting networks are a classical formal model for data-oblivious algorithms [9], where n inputs are fed into networks of n channels connected pairwise by comparators. Each comparator takes the two inputs from its two channels, compares them, and outputs them sorted back to the same two channels. A set of consecutive comparators can be viewed as a "parallel layer" if no two comparators act on the same channel. A comparator network is a sorting network if the output on the n channels is always the sorted sequence of the inputs.

Ever since sorting networks were introduced, there has been a quest to find optimal sorting networks: optimal size (minimal number of comparators), as well as optimal depth (minimal number of layers) networks. In their celebrated result, Ajtai, Komlós and Szemerédi [1] give a construction for sorting networks with $O(n \log n)$ comparators in $O(\log n)$ parallel levels. These AKS

*Supported by the Israel Science Foundation, grant 182/13 and by the Danish Council for Independent Research, Natural Sciences. Computational resources provided by an IBM Shared University Award (Israel).

sorting networks are a classical example of an algorithm optimal in theory, but highly inefficient in practice. Although they attain the theoretically optimal $O(n \log n)$ number of comparisons and $O(\log n)$ depth, the AKS networks are infamous for the large constants hidden in the big-O notation. On the other hand, already in 1968, Batcher [2] gave a simple recursive construction that, even though it creates networks of depth $O(\log^2 n)$, is superior to AKS networks for all practical values of n.

It is of particular interest to construct optimal sorting networks (both in size and in depth) for specific small numbers of inputs. Such networks can be used as building blocks to construct more efficient networks on larger numbers of inputs, for example by serving as base cases in recursive constructions such as Batcher's odd-even construction.

Already in the fifties and sixties various constructions appeared for small sorting networks on few inputs. In the 1973 edition of *"The Art of Computer Programming"* [9] (vol. 3, Section 5.3.4), Knuth detailed the smallest sorting networks known at the time with $n \leq 16$ inputs.

However, showing their optimality has proved to be extremely challenging. For $n \leq 8$ inputs, optimality was established by Knuth and Floyd [8] in 1973. No further progress had been made on the problem until 1989, when Parberry [17] showed that the networks given for $n = 9$ and $n = 10$ are also optimal. Parberry obtained this result by implementing an exhaustive search with pruning based on symmetries in the first two parallel steps in the sorting networks, and executing the algorithm on a Cray-2 supercomputer. Despite the great increase in available computational power in the two and a half decades since, his algorithm would still not be able to handle the case $n = 11$ or bigger. More recently, there were additional attempts [16] at solving the $n = 11$ case, but we are not aware of any successful one.

In this paper, some 40 years after the publication of the networks by Knuth, we finally prove their optimality by settling the remaining open cases of $11 \leq n \leq 16$ inputs. Our approach combines two methodologies: symmetry breaking and Boolean satisfiability.

Symmetry Breaking We show how to construct a small set R_n of two-layer networks on n channels such that: if there is a sorting network on n channels of a given depth, then there is one whose first two layers are in this set. We first show how each two-layer network can be represented by a graph with isomorphic graphs corresponding to equivalent networks. By defining a notion of "relative strength" between networks that takes into account their effects on the inputs, we further restrict the set of two-layer networks. We show how to characterize the strongest networks using context-free grammars, which enables us to construct the sets R_n for up to $n = 40$ inputs within two hours of computation. For example, R_{11} consists of 28 networks, enabling us to solve the optimal-depth problem for $n = 11$ in terms of only 28 independent cases as opposed to over one billion cases of all two-layer networks on 11 channels. Similarly, we show that $|R_{13}| = 117$.

Boolean Satisfiability With the first two layers restricted to a small set, we construct a family of propositional formulas whose satisfiability is necessary for the existence of sorting networks of a given size. Using an off-the-shelf SAT solver we show that all the constructed formulas are unsatisfiable, and hence we conclude that for $n \leq 16$ inputs the networks listed in [9] are indeed optimal. A similar construction, without restricting the first two layers, is able to find optimal-depth sorting networks for $n \leq 10$ inputs and prove them optimal, thus providing independent confirmation of the previously known results.

We obtained all our results using an off-the-shelf SAT solver running under Linux on commodity hardware. It is noteworthy that our algorithm required a few seconds to prove the optimality of networks with $n \leq 10$ inputs, whereas for $n = 10$ the algorithm described in [17] was estimated to take hundreds of hours on a supercomputer, and the algorithm described in [16] took more than three weeks on a desktop computer.

This paper is an extended version of [3] and [7]. The first paper presents the theory and experiments for calculating optimal sorting networks. In the current paper we construct even smaller sets of "non-isomorphic" two-layer networks using a much faster algorithm (the construction in [3] does not scale beyond $n = 13$ inputs). This new algorithm is a culmination of the work presented

in the second paper [7]. However, that paper deals only with computing the sets of "relevant" two-layer networks, and not with computing the optimal sorting networks as we do in this paper.

2 Preliminaries on sorting networks

An example of a comparator network on 4 channels is shown in Figure 1. The figure introduces the graphical notation used throughout the paper to depict comparator networks. Channels are indicated as horizontal lines (with channel 4 at the bottom), comparators are indicated as vertical lines connecting a pair of channels, and layers are separated by dashed lines. The figure further shows how the inputs $\langle 5, 2, 0, 7\rangle$ and $\langle 0, 1, 0, 1\rangle$ propagate from left to right through the network.

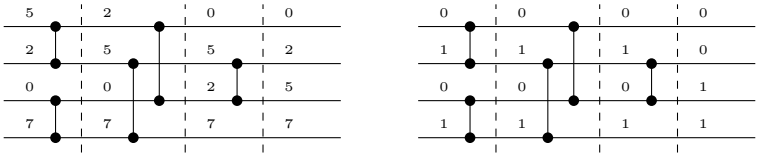

Figure 1: The comparator network $\{(1,2),(3,4)\};\{(1,3),(2,4)\};\{(2,3)\}$ with 4 channels, 5 comparators, and depth 3. On the left, the input $\langle 5, 2, 0, 7\rangle$ propagates through the network to give the output $\langle 0, 2, 5, 7\rangle$; on the right, the input $\langle 0, 1, 0, 1\rangle$ propagates through the network to give the output $\langle 0, 0, 1, 1\rangle$.

Formally, a *comparator network* C with n *channels* and *depth* d is a sequence $C = L_1; \ldots; L_d$ of d *layers*. Each layer L_k is a set of comparators (i, j), joining the channels i and j, with $1 \le i < j \le n$. In every layer L_k, each channel i is used by at most one comparator, i.e.

$$\left|\left\{\ j \;\middle|\; (i,j) \in L_k \vee (j,i) \in L_k\ \right\}\right| \le 1$$

for each i. The *size* of C is the total number of comparators in all its layers. Given comparator networks C_1 and C_2, let $C_1; C_2$ denote the comparator network obtained by concatenating the layers of C_1 and C_2. If C_1 has m layers, then it is an *m-layer prefix* of $C_1; C_2$.

An input to C is a sequence $\bar{x} = \langle x_1, \ldots, x_n\rangle$ of numbers. The input is propagated through the network, each comparator (i, j) outputting the smaller of its inputs on channel i and the larger of the inputs on channel j.

Denote by $C(\bar{x}, k, i)$ the value of channel $1 \le i \le n$ at layer $0 \le k \le d$ given input $\bar{x}$. Then we take $C(\bar{x}, 0, i) = x_i$ (input), and for $0 \le k < d$ we define:

$$C(\bar{x}, k+1, i) = \begin{cases} \min(C(\bar{x}, k, i), C(\bar{x}, k, j)) & \text{if } (i,j) \in L_{k+1} \\ \max(C(\bar{x}, k, i), C(\bar{x}, k, j)) & \text{if } (j,i) \in L_{k+1} \\ C(\bar{x}, k, i) & \text{otherwise.} \end{cases}$$

The *output* of C on $\bar{x}$ is the sequence $C(\bar{x}) = \langle C(\bar{x}, d, 1), C(\bar{x}, d, 2), \ldots, C(\bar{x}, d, n)\rangle$. A comparator network is called a *sorting network* if the output $C(\bar{x})$ is sorted (ascendingly) for all input sequences $\bar{x}$. The comparator network depicted in Figure 1 is a sorting network. The figure further indicates the values $C(\bar{x}, k, i)$ on each channel i after each layer k for $\bar{x} = \langle 5, 2, 0, 7\rangle$ (on the left) and for $\bar{x} = \langle 0, 0, 1, 1\rangle$ (on the right).

Given sufficient parallel computational power (e.g., assuming the networks are directly implemented in hardware), independent comparators can be evaluated in parallel, and hence the depth of a sorting network corresponds to the number of parallel steps needed to sort n inputs. Thus, given number of channels n, we focus on finding sorting networks of minimal depth.[1] We denote the smallest depth of a sorting network on n channels by $T(n)$.

[1] In general, it is possible to construct a sorting network with fewer comparators, albeit larger depth, than the one that achieves $T(n)$. We refer the readers interested in the minimum number of comparators needed to sort n channels to [9] and [6] where the optimal values are presented for $n \le 8$ and $n = 9, 10$, respectively.

Prior to this paper, the precise values of $T(n)$ were known only for $n \leq 10$: the values for $n \leq 8$ are given in [9], and those for $n = 9, 10$ are reported by Parberry in [18]. These, and the best previously known bounds for $n \leq 16$, are summarized in Table 1.

n	1	2	3,4	5,6	7,8	9,10	11,12	13,14,15,16
$T(n) \leq$	0	1	3	5	6	7	8	9
$T(n) \geq$	0	1	3	5	6	7	7	7

Table 1: The best previously known upper and lower bounds for $T(n)$.

In this paper, we prove optimality of the upper bounds of $T(n)$ for $11 \leq n \leq 16$. For example, we show that $T(11) \geq 8$. To prove such a result, we have to establish that none of the 7-layer, 11-channel comparator networks is a sorting network.

Each comparator joins two distinct channels, and hence one can view each layer of an n-channel comparator network as a matching on n elements [18]. It turns out that there are 35,696 matchings on 11 elements.[2] So, to establish the lower bound $T(11) \geq 8$, we have to show that none of the $35{,}696^7 \geq 10^{31}$ comparator networks on 11 channels with 7 layers is a sorting network. Similarly, establishing that $T(13) \geq 9$ requires showing that none of the $568{,}504^8 \geq 10^{46}$ comparator networks on 13 channels with 8 layers is a sorting network. These numbers immediately make any form of exhaustive search infeasible.

In the first part of this paper, we show how to reduce the size of this search space. Then, in Section 5, we reduce the existence of a sorting network to the problem of propositional satisfiability.

In full, our algorithm to determine whether a sorting network of a given depth exists consists of four phases. In the first phase, we extend the approach introduced by Parberry in [18], and partition of the set of two-layer comparator networks into equivalence classes. We then select representatives of some of these equivalence classes so that, if there is a sorting network of the given size, then there is one with the first two layers equal to some chosen representative. In the next phase, we reduce the existence of a sorting network beginning with one of the calculated representatives to satisfiability of a corresponding propositional formula. Finally, we determine the satisfiability of the obtained formulas using a SAT solver.

On the face of it, to determine whether a given n-channel candidate comparator network is a sorting one it seems necessary to try all possible permutations of $\{1, \ldots, n\}$ as inputs. The following classical result states that it suffices to consider only Boolean inputs, i.e. sequences of 0 and 1s. This reduces the size of the set of inputs from $n!$ permutations to 2^n Boolean inputs.

Lemma 1 (The zero-one principle [9])**.** *A comparator network C is a sorting network if and only if C sorts all Boolean inputs.*

In the remainder of the paper, we consider only comparator networks with Boolean inputs. We will write $\mathcal{B}^n = \{0, 1\}^n$ to denote the set of Boolean inputs, and, given a comparator network C, we define $\mathsf{outputs}(C) = \left\{ \; C(\bar{x}) \;\middle|\; \bar{x} \in \mathcal{B}^n \; \right\}$. Hence, C is a sorting network if and only if all elements of $\mathsf{outputs}(C)$ are sorted (in ascending order).

3 Equivalence of comparator networks

A first step in reducing the search space of all comparator networks can already be found in the work of Parberry [18]. A layer on n channels is called *maximal* if it contains $\left\lfloor \frac{n}{2} \right\rfloor$ comparators, i.e., no further comparators can be added to the layer.

Lemma 2 (Parberry [18])**.** *Let L be any maximal layer on n channels. If there is a sorting network on n channels with depth d, then there is one whose first layer is L.*

[2] Sequence `A000085` of the the On-Line Encyclopedia of Integer Sequences, published electronically at `http://oeis.org`.

This lemma implies that, when searching for an optimal-depth sorting network, the first layer can be fixed to any maximal first layer, effectively reducing the problem by one layer. In this paper we consider the following choice of the first layer of n-channel sorting networks:

$$F_n = \left\{ \; (2i-1, 2i) \;\middle|\; 1 \le i \le \left\lfloor \tfrac{n}{2} \right\rfloor \; \right\}$$

See for example the network of Figure 1, whose first layer is F_4.

Assuming the first layer has been fixed to some maximal layer, Parberry also states [18] that one need not consider second layers which are identical modulo permutations of channels. Let π be a permutation on $\{1, \ldots, n\}$. For a comparator (i, j) we define $\pi((i, j)) = (\pi(i), \pi(j))$. Then for a layer L, we define

$$\pi(L) = \left\{ \; (\pi(i), \pi(j)) \;\middle|\; (i, j) \in L \; \right\}.$$

If $i < j$ for each $(i, j) \in \pi(L)$, then $\pi(L)$ is also a layer, otherwise we say that $\pi(L)$ is a *generalized layer*. The definition naturally extends to a network $C = L_1; \cdots ; L_k$ by applying the permutation on each layer independently: $\pi(C) = \pi(L_1); \cdots ; \pi(L_k)$, yielding a *generalized comparator network*. It is well known (see e.g. Exercise 5.3.4.16 of [9]) that a generalized sorting network can always be "untangled" into a standard sorting network of the same dimensions (see Figure 2 for an example). Furthermore, this operation preserves the "standard prefix", i.e., the longest prefix of the network that does not have generalized layers. Formally, if E is a comparator network and F is a generalized comparator network such that $E; F$ is a generalized sorting network, then there is a comparator network F' of the same depth as F so that $E; F'$ is a sorting network.

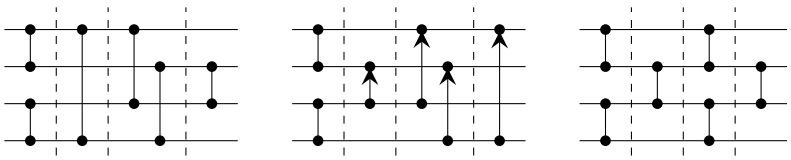

Figure 2: The 4-channel networks to the left and to the right are equivalent via the permutation $(1\,3)(2\,4)$. The middle network is the generalized comparator network obtained by applying the permutation to the left comparator network. The right network can then be obtained by untangling the middle network.

Taking into account that the first layer is fixed (Lemma 2), Parberry [18] considered permutations that leave the first layer intact.

Lemma 3 (Theorem 5.4 of [18])**.** *Let L_1 and L_2 be layers on n channels. Let π be a permutation such that L_1 is maximal, $\pi(L_1) = L_1$, and $\pi(L_2)$ is a layer. Then there is a depth-d sorting network of the form $L_1; L_2; C$ if and only if there is one of the form $L_1; \pi(L_2); C'$.*

The proof of depth optimality for sorting networks with 9 channels described in [18] is based on the application of Lemma 3 together with a brute force algorithm that first partitions the set of two-layer networks with a fixed maximal first layer into equivalence classes modulo permutations that fix the first layer. The "small" number of equivalence classes for $n \le 10$ channels is computed in this way and reported in [18]. However, when partitioning networks into equivalence classes using a brute-force approach, one must consider the rapidly increasing number of permutations, and this approach does not scale as the number of channels grows. Furthermore, even if these equivalence classes were given, the search algorithm described in [18] does not scale for larger numbers of channels.

The main theme of the first half of this paper is a better computation and exploitation of symmetries in the first two levels of comparator networks. Using the terminology of the definition below, we aim to find as small as possible *complete sets of filters*. For example, for $n = 16$ we reduce the number of second layers that must be considered from 46,206,736 to only 211.

Definition 1. *A set $\mathcal{F}$ of comparator networks on n channels is a complete set of filters for the optimal-depth sorting network problem if there exists an optimal-depth sorting network on n channels of the form $C; C'$ for some $C \in \mathcal{F}$.*

We now introduce a notion of equivalence of comparator networks that is stronger than the one considered by Parberry [18]. Let C be a comparator network on n channels. The graph representation of C is a directed and labeled graph $\mathcal{G}(C) = (V, E)$, where each node in V corresponds to a comparator in C and $E \subseteq V \times \{1, 2\} \times V$. Let $c(v)$ denote the comparator corresponding to a node v. Then $(u, 1, v) \in E$ if the minimum output of $c(u)$ is an input of $c(v)$, and $(u, 2, v) \in E$ if the maximum output of $c(u)$ is an input of $c(v)$. Note that the number of channels cannot be inferred from this representation, as channels that are unused are not represented.

Figure 3 illustrates the graph representations of the left and right networks from Figure 2, where the comparators are labeled alphabetically in order of occurrence (left-to-right, top-down). Note that these two graphs can be seen to be isomorphic by mapping the vertices as $a \mapsto b'$, $b \mapsto a'$, $c \mapsto c'$, $d \mapsto d'$, $e \mapsto e'$ and $f \mapsto f'$.

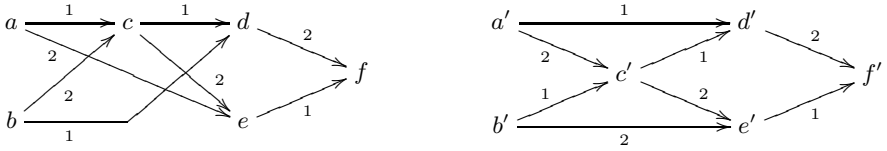

Figure 3: Graph representations of the two networks in Figure 2.

Clearly, graphs representing comparator networks are acyclic, and the degrees of their vertices are bounded by 4. The strong relationship between equivalence of comparator networks and isomorphism of their corresponding graphs is reflected in Lemma 4 below, which implies that the comparator network equivalence problem is polynomially reduced to the bounded-valence graph isomorphism problem.

Definition 2. *Let C_1 and C_2 be n-channel comparator networks. Then we write $C_1 \approx C_2$ if $\mathcal{G}(C_1)$ and $\mathcal{G}(C_2)$ are isomorphic.*

Lemma 4 (Choi & Moon [4], Proposition 2)**.** *Let $C = C_1; D$ and C_2 be n-channel comparator networks such that C_1 and C_2 have the same depth and $\mathcal{G}(C_1) \approx \mathcal{G}(C_2)$. If C is a sorting network, then there is an n-channel comparator network D' of the same depth as D such that $C_2; D'$ is a sorting network.*

The graph isomorphism problem is one of a very small number of problems belonging to NP that are neither known to be solvable in polynomial time nor known to be NP-complete. However, it is known that the isomorphism of graphs of bounded valence can be tested in polynomial time [10], so the comparator network equivalence problem can be efficiently solved.

4 Complete sets of two-layer filters

Recall that our goal is for given n to compute as small as possible complete sets of filters consisting of two-layer networks on n channels. We now show how to exploit the graph representation to compute such a set.

4.1 A Symbolic Representation

The obvious approach for finding all two-layer prefixes modulo symmetry is to generate all two-layer networks, and then apply graph isomorphism to find canonical representatives of the equivalence classes. We evaluated this approach using the popular graph isomorphism tool `nauty` [11], but found that the exponential growth in the number of two-layer prefixes prevents this approach from scaling. Therefore, we opted for a symbolic representation of these graphs that captures isomorphism.

For the special case of two-layer networks, the vertices in the resulting graphs always have degree 1 or 2. Therefore, they always consist of sets of "sticks" and "cycles", and they are completely

characterized by the maximal-length simple paths they contain. Moreover, this representation can be determined directly from the network, as illustrated in Figure 4.

It is useful to adopt the following terminology on channels. A channel in a comparator network is called a *min-channel* (respectively, a *max-channel*) if it is the smaller (resp. larger) channel in some comparator of the first layer. We will also occasionally refer to a min- or max- channel at a layer d with the obvious meaning. A channel of a comparator network is called a *free* channel if it is not used in the first layer.

Definition 3. *A* path *in a two-layer network C is a sequence $\langle p_1 p_2 \dots p_k \rangle$ of distinct channels such that each pair of consecutive channels is connected by a comparator in C.*

The word *corresponding to $\langle p_1 p_2 \dots p_k \rangle$ is $\langle w_1 w_2 \dots w_k \rangle$, where:*

$$w_i = \begin{cases} 0 & \text{if } p_i \text{ is the free channel} \\ 1 & \text{if } p_i \text{ is a min-channel} \\ 2 & \text{if } p_i \text{ is a max-channel} \end{cases}$$

A path is maximal if it is a simple path (with no repeated nodes) that cannot be extended (in either direction). A network is connected if its graph representation is connected.

Definition 4. *Let C be a connected two-layer network on n channels. Then $\mathsf{word}(C)$ is defined as follows, where there are three kinds of words.*

Head-word. *If n is odd, then $\mathsf{word}(C)$ is the word corresponding to the maximal path in C starting with the (unique) free channel.*

Stick-word. *If n is even and C has two channels not used in layer 2, then $\mathsf{word}(C)$ is the lexicographically smallest of the words corresponding to the two maximal paths in C starting with one of these unused channels (which are reverse to one another).*

Cycle-word. *If n is even and all channels are used by a comparator in layer 2, then $\mathsf{word}(C)$ is the lexicographically smallest word corresponding to a maximal path in C that begins with two channels connected in layer 1.*

The set of all possible words (not necessarily minimal with respect to lexicographic ordering) can be described by the following BNF-style grammar.

$$\begin{aligned} &\mathsf{Word} ::= \mathsf{Head} \mid \mathsf{Stick} \mid \mathsf{Cycle} && \mathsf{Stick} ::= (\mathtt{12} + \mathtt{21})^+ \\ &\mathsf{Head} ::= \mathtt{0}(\mathtt{12} + \mathtt{21})^* && \mathsf{Cycle} ::= \mathtt{12}(\mathtt{12} + \mathtt{21})^+ \end{aligned} \tag{1}$$

To avoid ambiguity, we annotate each word with a tag from the set $\{\mathtt{h}, \mathtt{s}, \mathtt{c}\}$ to indicate whether it is a Head-, Stick- or Cycle-word, respectively. In Figure 4, the three two-layer networks a–c lead to the generation of a word of each kind resulting from the paths shown in a'–c', respectively.

Definition 5. *The* word representation *of a two-layer comparator network C, $\mathsf{word}(C)$, is the multi-set containing $\mathsf{word}(C')$ for each connected component C' of C; we will denote this set by the "sentence" $w_1; w_2; \dots; w_k$, where the words are in lexicographic order (including their tags).*

In particular, a connected network will be represented by a sentence with only one word, so there is no ambiguity in the notation $\mathsf{word}(C)$. The requirement that the first layer is maximal corresponds to the requirement that the multi-set $\mathsf{word}(C)$ has at most one Head-word. Figure 4(d) illustrates the case of a multi-component two-layer network.

Conversely, a word w defines a two-layer network as follows.

Definition 6. *Let w be a word in the language of Equation (1), and $n = |w|$. The two-layer network $\mathsf{net}(w)$ has first layer F_n and second layer defined as follows.*

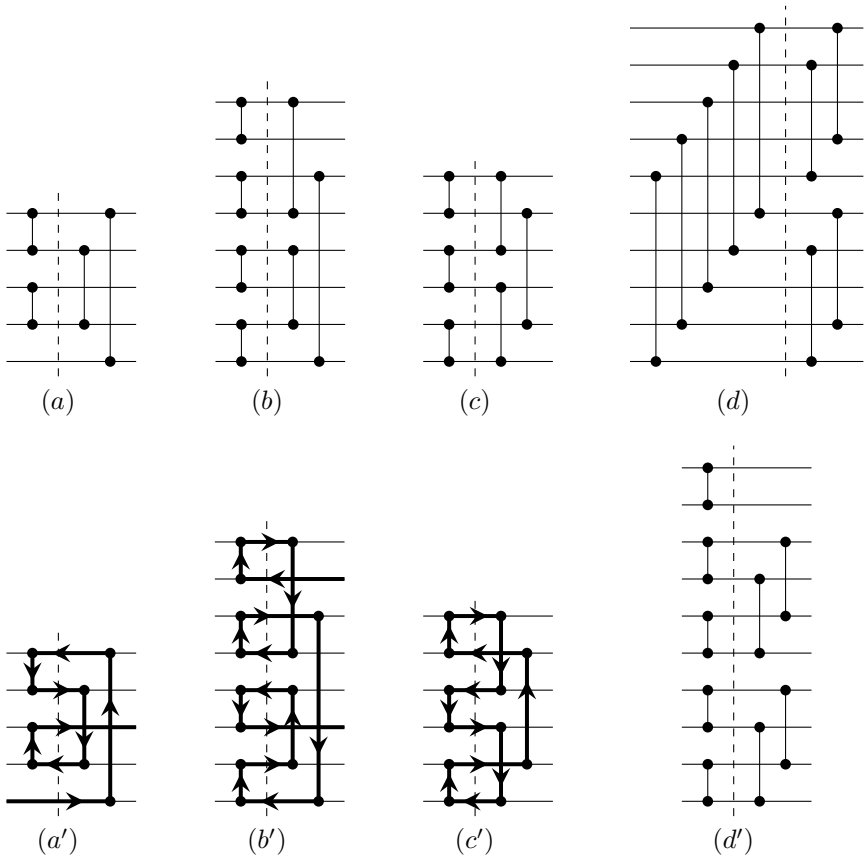

Figure 4: Networks and paths. Networks (a–c) correspond to the three cases in Definition 4, and (a'–c') depict the corresponding maximal paths. Path (a') starts from the free channel, with corresponding word $\texttt{01221}_\texttt{h}$. Path ($b'$) starts from a free channel at layer 2, corresponding to the word $\texttt{21212112}_\texttt{s}$; the reverse path corresponds to the smaller word $\texttt{21121212}_\texttt{s}$, which represents network (b). For the cycle in (c'), the smallest word is $\texttt{121221}_\texttt{c}$, obtained on the reverse path starting from channel 1.
Network (d) consists of three connected components (the sets of channels $\{1, 4, 6, 9\}$, $\{2, 5, 7, 10\}$ and $\{3, 8\}$), corresponding to the first two layers of the 10-channel sorting network from Figure 49 of [9]. The first two components contain cycles represented by $\texttt{1221}_\texttt{c}$, and the third yields the Stick-word $\texttt{12}_\texttt{s}$. The network is thus represented by the sentence $\texttt{12}_\texttt{s}; \texttt{1221}_\texttt{c}; \texttt{1221}_\texttt{c}$. In turn, this sentence generates the equivalent network (d').

n	3	4	5	6	7	8	9	10	11	12	13	14
$\lvert G_n\rvert$	4	10	26	76	232	764	2,620	9,496	35,696	140,152	568,504	2,390,480
$\lvert R(G_n)\rvert$	4	8	16	20	52	61	165	152	482	414	1,378	1,024

n	15	16	17	18	19
$\lvert G_n\rvert$	10,349,536	46,206,736	211,799,312	997,313,824	4,809,701,440
$\lvert R(G_n)\rvert$	3,780	2,627	10,187	6,422	26,796

Table 2: Values of $|G_n|$ and $|R(G_n)|$ for $n \leq 19$. Besides the values given in the table, $|R(G_{20})| = 15{,}906$ was computed in a few seconds, and $|R(G_{30})| = 1{,}248{,}696$ in under a minute.

1. *If w is a Stick-word or a Cycle-word, ignore the first character; then, for $k = 0, \ldots, \lfloor \frac{n}{2} \rfloor - 1$, take the next two characters xy of w and add a second-layer comparator between channels $2k + x$ and $2(k + 1) + y$. Ignore the last character; if w is a Cycle-word, connect the two remaining channels at the end.*

2. *If w is a Head-word, proceed as above but start by connecting the free channel to the channel indicated by the second character.*

To generate a network from a sentence, we simply generate the networks for each word in the sentence and compose them in the same order. Figure 4(d') illustrates this construction.

The following lemma shows that abstracting networks to words captures network equivalence.

Lemma 5. *Let C and C' be two-layer comparator networks on n channels. Then $C \approx C'$ if and only if* $\mathsf{word}(C) = \mathsf{word}(C')$.

Proof. The "if" part follows from the observation that, for two-layer networks, $C \approx C'$ means that there is a permutation π such that C' equals $\pi(C)$ with possibly some comparators reversed in the second layer. Thus, any path obtained in C beginning at channel j can be obtained in C' by beginning at channel $\pi(j)$, and vice versa. The "only if" part is straightforward. □

Remark 1. *In algebraic terms, the functions* word *and* net *form an adjunction between the preorders of words (with lexicographic ordering) and two-layer comparator networks (with equivalence). The function* word *can be seen as a "forgetful" functor that forgets the specific order of channels in a net, whereas* net *generates the "free" network with first layer F_n from a given word. Furthermore,* word *always returns the minimum element in the fiber* $\mathsf{net}^{-1}(w)$*, whence lexicographic minimal words can be used to characterize equivalent networks.*

Remark 2. *Note that Definition 6 can be adapted to any choice of a maximal first layer.*

Definition 6 fixes the first layer of a two-layer comparator network $\mathsf{net}(w)$ to F_n. Denote the set of all possible second layers (in a two-layer network whose first layer is F_n) by G_n. We denote the equivalence classes of the two-layer networks whose first layer is F_n and whose second layer is a member of G_n by $R(G_n)$. We view $R(G_n)$ as a set of representatives of the equivalence classes. So $R(G_n)$ is viewed as a maximal set of non-equivalent networks.

Theorem 1. *For any $n \geq 3$, the set $R(G_n)$ of two-layer comparator networks is a complete set of filters for the optimal-depth sorting network problem.*

As a consequence of Lemma 5, $R(G_n)$ can be constructed simply by generating all multi-sets of words with at most one Head-word yielding exactly n channels. This procedure has been implemented straightforwardly in Prolog, resulting in the values in Table 2.

As mentioned in Section 2, $|G_n|$ corresponds to the number of matchings in a complete graph with n nodes, since every comparator joins two channels. The sequence $|R(G_n)|$ does not appear to be known already, and it does not appear to have a simple description. The following property is interesting to note.

Theorem 2. *For odd n, $|R(G_n)| = |R(G_{n-1})| + 2\,|R(G_{n-2})|$.*

Proof. The proof is based on the word representation of the networks. If n is odd, then $\mathsf{word}(C)$ contains exactly one word beginning with $\mathtt{0}$. If this word is $\mathtt{0_h}$, then removing it yields a network with $n-1$ channels, and this construction is reversible. Otherwise, removing the two last letters in this word yields a network with $n-2$ channels; since the removed letters can be $\mathtt{12}$ or $\mathtt{21}$, this matches each network on $n-2$ channels to two networks on n channels. □

4.2 Saturation

We now introduce a notion that further restricts the set of networks we need to consider when searching for optimal sorting networks. Instead of just looking at the structure of the comparators (modulo permutation), we further take into account the actual effect of the network on its inputs, and focus only on those networks that achieve the "most" amount of sorting. Similar to the way that we use grammars to characterize isomorphic networks, here too we first define the desired semantic property, and later provide a syntactic characterization in terms of a grammar. The following lemma makes precise what we mean by achieving the "most" amount of sorting.

Lemma 6. *Let $C = P; S$ be a sorting network of depth d and Q be a comparator network such that P and Q have the same depth, and $\mathsf{outputs}(Q) \subseteq \mathsf{outputs}(P)$. Then $Q; S$ is a sorting network of depth d.*

Proof. Since P and Q have the same depth, the depth of $Q; S$ is d. Let $\bar{x} \in \mathcal{B}^n$ be an arbitrary input. Then $Q(x) \in \mathsf{outputs}(Q) \subseteq \mathsf{outputs}(P)$. Hence, there is $y \in \mathcal{B}^n$ such that $Q(x) = P(y)$. Thus, $(Q;S)(x) = S(Q(x)) = S(P(y)) = (P;S)(y) = C(y)$, which is sorted since C is a sorting network. □

Lemma 6 generalizes, so that it suffices that there exists a permutation mapping the set of outputs of one network into the set of outputs of the other network.

Lemma 7. *Let $C = P; S$ be a sorting network of depth d and Q be a comparator network such that P and Q have the same depth, and $\mathsf{outputs}(Q) \subseteq \pi(\mathsf{outputs}(P))$ for some permutation π on n channels. Then there exists a sorting network of the form $Q; S'$ of depth d.*

Proof. Let $C = P; S$ be a sorting network of depth d. Then $\pi(C) = \pi(P); \pi(S)$ is a generalized sorting network. Since $\mathsf{outputs}(Q) \subseteq \pi(\mathsf{outputs}(P)) = \mathsf{outputs}(\pi(P))$, Lemma 6 implies that $Q; \pi(S)$ is also a generalized sorting network. Untangling $\pi(S)$, we obtain S' such that $Q; S'$ is a sorting network of depth d. □

For comparator networks C_a and C_b, if $\mathsf{outputs}(C_b) \subseteq \pi(\mathsf{outputs}(C_a))$ for some permutation π, we write $C_b \preceq C_a$, and say that C_b *subsumes* C_a. Note that this relation includes equivalence. By Lemma 7, it suffices to consider two-layer networks that are minimal with respect to subsumption.

Corollary 1. *The set of equivalence classes of two-layer networks that are minimal with respect to subsumption is a complete set of filters.*

Suppose one wishes to compute these minimal (up to the subsumes relation) elements directly. Having fixed the first layer to some maximal layer (e.g., F_n), there are still $|G_n|$ many possibilities for the second layer (see Table 2), the size of their output sets is potentially exponential, and there are $n!$ permutations to consider, per pair of output sets, to determine subsumption. So this problem quickly becomes intractable. One might consider clever optimizations to reduce the computation time, but such an approach does not scale well either.

Instead, we shall define a new class of networks, which we call *saturated two-layer networks*. This class, as it turns out, has a simple syntactical characterization using the notion of words. Moreover, we shall prove that it forms a complete set of filters. We experimentally verify for small values of n that the class of saturated two-layer networks is precisely the set of minimal two-layer networks with respect to subsumption. We conjecture that the equality holds for all values of n.

Note that our results on the depth-optimality of sorting networks do not depend on this conjecture as we only require that the class of saturated two-layer prefixes forms a complete set of filters, which we prove in Theorem 6.

Before we introduce saturated networks formally, we point out that restricting attention to such networks significantly reduces the number of two-layer prefixes we need to consider. The numbers $|S_n|$ of saturated two-layer networks and $|R(S_n)|$ of their equivalence classes modulo graph isomorphism are given in Table 3. For example, for $n = 16$, we reduce the number of two-layer prefixes to consider from 2,627 to 323.

Definition 7. *A comparator network C is* redundant *if there exists a network C' obtained from C by removing a comparator such that* $\mathsf{outputs}(C') = \pi(\mathsf{outputs}(C))$ *for some permutation π. A network C is* saturated *if it is non redundant, and every network C' obtained by adding a comparator to C satisfies* $\mathsf{outputs}(C') \not\subseteq \pi(\mathsf{outputs}(C))$ *for every permutation π.*

For example, any comparator network that contains comparators between the same two channels at consecutive layers is redundant. The notion of saturation is a generalization of Parberry's work in the first layer [17]: Lemma 2 can be restated as saying that the first layer of a saturated comparator network on n channels always contains $\lfloor \frac{n}{2} \rfloor$ comparators.

The following property quantifies the impact of removing redundant two-layer prefixes.

Theorem 3. *The number of non-equivalent redundant two-layer networks on n channels is $|R(G_{n-2})|$.*

Proof. The proof is based on the word representation of the networks. If C is a redundant net, then the sentence $\mathsf{word}(C)$ contains the word $\mathtt{12_c}$. Removing one occurrence of this word yields a sentence corresponding to a network with $n-2$ channels. This construction is reversible, so there are $|R(G_{n-2})|$ words corresponding to redundant networks on n channels. □

In order to characterize saturated networks syntactically, we adopt the notion of a pattern. A *pattern* P is a partially specified network: it is a set of channels connected by comparators, but it may also include "external" comparators (represented as a singleton node) that are connected to one channel in P and one channel not in P. A comparator network C contains a pattern P of depth d on m channels if there are a depth-d prefix C_1 of C and distinct channels $c_1, \ldots, c_m$ of C_1 such that: (i) if P contains a comparator between channels i and j at layer $1 \leq k \leq d$, then C_1 contains a comparator between channels c_i and c_j at layer k; (ii) if P contains an external comparator touching channel i at layer $1 \leq k \leq d$, then C_1 contains a comparator between channel c_i and a channel $c \notin \{c_1, \ldots, c_m\}$ at layer k; (iii) C_1 contains no other comparators connecting to or between channels $c_1, \ldots, c_m$.

Figure 5 depicts two patterns (a) and (b) and two networks (c) and (d). The depth 2, 3-channel pattern depicted in (a) occurs in network (c) but not in (d), while the pattern in (b) does not occur in either network (c) or (d): its third channel is never used, while all channels of (c) and (d) are used in the first two layers.

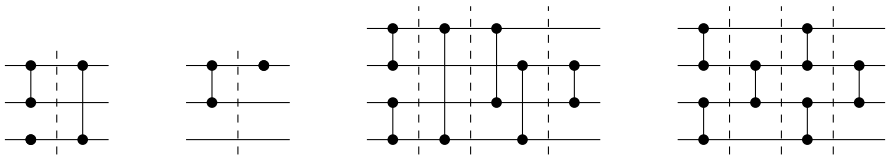

Figure 5: Two patterns (a, b) and two networks (c, d).

Theorem 4. *Let C be a saturated two-layer network. Then C contains none of the two-layer patterns in Figure 6.*

Proof. In all cases we show how to find i and j such that $\mathsf{outputs}(C;(i,j)) \subseteq \mathsf{outputs}(C)$ whenever C contains one of the patterns in Figure 6. Depending on how the pattern is embedded in C, (i,j) may be a generalized comparator; in that case, $\mathsf{outputs}(C;(j,i)) \subseteq \mathsf{outputs}(\pi(C))$ for $\pi = (i\,j)$.

1. Assume by contradiction that C includes the pattern $(1a)$ and let the channels corresponding to those in the pattern be a, b and c, from top to bottom. Define $C' = C;(c,b)$ and let $\bar{x} \in \mathcal{B}^n$. If $C'(\bar{x}) \neq C(\bar{x})$, then $C(\bar{x},1,c) = 1$ and $C(\bar{x},1,b) = 0$. Since b is a max-channel, this means that $C(\bar{x},0,a) = C(\bar{x},0,b) = 0$ and $C(\bar{x},0,c) = 1$. Then $C(\bar{x}') = C'(\bar{x})$ for the input $\bar{x}'$ obtained from $\bar{x}$ by exchanging the values in positions c and b. Therefore $\mathsf{outputs}(C') \subseteq \mathsf{outputs}(C)$, contradicting the fact that C is saturated.

   Case $(1b)$ is similar, adding a comparator (a,c), and either construction applies to case $(1c)$.

2. Assume by contradiction that C includes the pattern (2), and let the channels corresponding to those in the pattern be a, b, c and d, from top to bottom. Define $C' = C;(a,d)$, and let $\bar{x} \in \mathcal{B}^n$. If $C'(\bar{x}) \neq C(\bar{x})$, then $C(\bar{x},1,a) = 1$ and $C(\bar{x},1,d) = 0$. Since a is a min-channel and d is a max-channel, this means that $C(\bar{x},0,a) = (\bar{x},0,b) = 1$ and $(\bar{x},0,c) = (\bar{x},0,d) = 0$. Then $C(\bar{x}') = C'(\bar{x})$ for the input $\bar{x}'$ obtained from $\bar{x}$ by exchanging the values in positions a and d. Therefore $\mathsf{outputs}(C') \subseteq \mathsf{outputs}(C)$, contradicting the fact that C is saturated.

3. Assume by contradiction that C includes the pattern $(3a)$, and let the channels corresponding to those in the pattern be a, b, c and d, from top to bottom. Define $C' = C;(b,d)$, and let $\bar{x} \in \mathcal{B}^n$. If $C'(\bar{x}) \neq C(\bar{x})$, then $C(\bar{x},1,b) = 1$ and $C(\bar{x},1,d) = 0$. Then $C(\bar{x}') = C'(\bar{x})$ for the input $\bar{x}'$ obtained from $\bar{x}$ by exchanging the values in positions a and b with the values in positions c and d, respectively. Note that this will permute $C(\bar{x},1,a)$ and $C(\bar{x},1,c)$, but it will not affect the final values on channels a and c. Therefore $\mathsf{outputs}(C') \subseteq \mathsf{outputs}(C)$, contradicting the fact that C is saturated.

   Case $(3b)$ is similar.

In all three cases, it is straightforward to verify that the inclusion $\mathsf{outputs}(C') \subseteq \mathsf{outputs}(C)$ is strict. □

In fact, the patterns in Figure 6 are actually *all* of the patterns that make a comparator network non-saturated.

Theorem 5. *If C is a non-redundant two-layer comparator network on n channels containing none of the patterns in Figure 6, then C is saturated.*

Proof. Let C be a non-redundant two-layer comparator network, and assume that the second layer of C has at least two unused channels (otherwise there is nothing to prove). If one of these channels were unused at layer 1, then the network would contain pattern $(1a)$, $(1b)$ or $(1c)$. Thus, the two channels are necessarily used in a comparator in layer 1 by Theorem 4. From the same theorem, they must be both min-channels or both max-channels, otherwise the network would contain pattern (2); and the channels they are connected to at layer 1 cannot be connected at layer 2, otherwise the network would contain pattern $(3a)$ or $(3b)$.

There are eight different cases to consider. We detail the cases where the two unused channels are max channels. Assume that the four relevant channels are adjacent, labeled a, b, c and d from

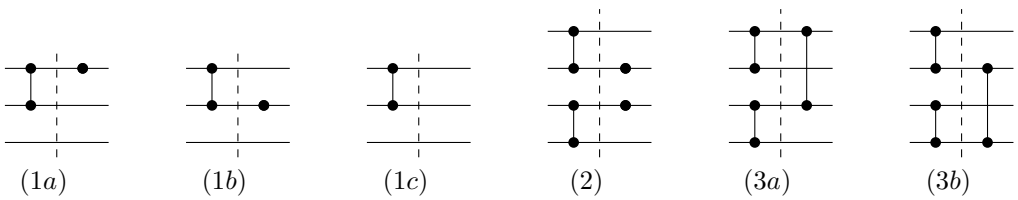

Figure 6: Patterns forbidden in a saturated two-layer network.

top to bottom, with first-layer comparators (a, b) and (c, d). This does not lose generality, but makes the presentation simpler: for the general case, just apply the permutation that brings any network to this particular form to the reasoning below. This transformation can always be done preserving the standard comparator network form.

Let k be the number of channels above a and m be the number of channels below d. Let C' be obtained from C by adding the comparator (b, d) at layer 2. The four possibilities depend on whether channels a and c are min- or max-channels at layer 2, and are represented in Figure 7.

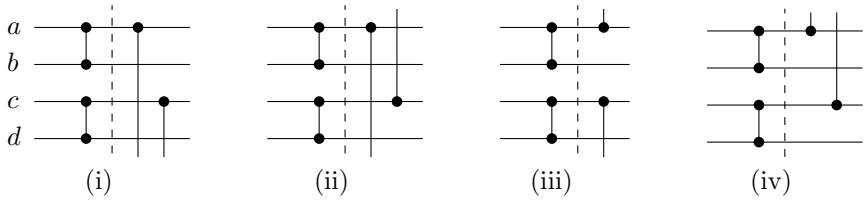

Figure 7: Possible cases for channels a and c in the proof of Theorem 5. To obtain C', add a comparator between channels b and d.

(Min/min) Channel a and b are min-channels at layer 2, so the network looks as in Case (i) of Figure 7.

Consider the input string 1^k11001^m. Since $C'(1^k11001^m) = 1^k10011^m$, we have $1^k10011^m \in \mathsf{outputs}(C')$. We now show that $1^k10011^m \notin \mathsf{outputs}(C)$. Because of the comparator (a, b) at layer 1, to obtain the 0 on channel b the input string would necessarily have a 0 on channel a. But then the output would also have a 0 on channel a, hence it could not be 1^k10011^m.

(Min/max) The network now looks as in Case (ii) of Figure 7, and the argument is similar. Choose an input $\bar{x}$ such that $C'(\bar{x})$ has 1001 on channels a–d; this is possible by placing 0s on the two positions that may be compared to c at layer 2 and 1s on the two positions that may be compared to a at layer 2. Then the argument from the previous case again shows that $C'(\bar{x}) \notin \mathsf{outputs}(C)$.

(Max/min) The network now looks as in Case (iii) of Figure 7. This can be reduced to the previous case by interchanging a and b with c and d, respectively.

(Max/max) The network now looks as in Case (iv) of Figure 7 and the reasoning is a bit more involved.

Consider once more the input string 1^k11001^m. Channel c is now a second-layer max-channel connected to some channel $j \leq k$, and $C'(1^k11001^m) = 1^{j-1}01^{k-j}10111^m$. In order to obtain this output with network C, it is again necessary to have inputs 0 on channels a and b; but since there are only two 0s in the output, this means that channel a must also be connected to channel j on layer 2, which is impossible.

The cases where a and c are the unused (min) channels are similar.

(Min/min) Similar to the case (Max/max) above, using the input string 0^k11000^m and analyzing the result on channel c.

(Min/max) Similar to the case (Min/max) above, using an input string that produces an output of the form $v1001w$, and analyzing the result on channel c.

(Max/min) This can be reduced to the previous case by interchanging a and b with c and d, respectively.

(Max/max) Similar to the case (Min/min) above, using the input string 0^k11000^m, and analyzing the result on channel c. □

As a corollary of Theorems 4 and 5, we show that we can always assume the first two layers of a sorting network to be saturated.

Corollary 2. *Let L_1 and L_2 be layers on n channels such that L_1 is maximal. Then there is a layer S such that* $\mathsf{outputs}(L_1; S) \subseteq \mathsf{outputs}(L_1; L_2)$ *and $L_1; S$ is saturated.*

Proof. By removing comparators if necessary, we can assume that $L_1; L_2$ is nonredundant. If $L_1; L_2$ is not saturated, then, by Theorem 5, it must contain some of the patterns from Figure 6. Now, for each pattern occurring in $L_1; L_2$, the argument in the proof of Theorem 4 tells us how to eliminate it by adding comparators to L_2. Denote the obtained layer by S. The argument in the proof of Theorem 4 further ensures that

By construction, the network $L_1; S$ does not contain any of the patterns from Figure 6, and so, by Theorem 5, it is saturated. □

The above Corollary together with Lemma 7 imply that if there is a sorting network of a given size, then there is one whose first two layers are saturated, i.e., the set of all saturated two-layer networks is a complete set of filters.

Theorem 6. *For every n, both the set S_n of two-layer saturated networks and the set of representatives of its equivalence classes $R(S_n)$ are complete sets of filters on n channels.*

Proof. Corollary 2 and Lemma 7 imply that S_n is a complete set of filters. By Lemma 4, $R(S_n)$ is also a complete set of filters. □

We conjecture that in fact the following result holds.

Conjecture 1. *If networks C_1 and C_2 on n channels are both saturated and non-equivalent, then* $\mathsf{outputs}(C_1) \not\subseteq (C_2)$ *for any permutation π.*

Particular cases of Conjecture 1 are implied by Theorem 5, but the general case remains open. The conjecture has been verified experimentally for $n \leq 15$.

The characterization of saturation given by Theorem 5 is straightforward to translate in terms of the word associated with a network.

Corollary 3. *Let C be a two-layer network. Then C is saturated if $w = \mathsf{word}(C)$ satisfies the following properties.*

1. *If w contains $\mathtt{0_h}$ or $\mathtt{12_s}$, then all other words in w are cycles.*
2. *No stick w has length 4.*
3. *Every stick in w begins and ends with the same symbol.*
4. *If w contains a head or stick ending with x, then every head or stick in w ends with x, for $x \in \{\mathtt{1}, \mathtt{2}\}$.*

Thus, the set of saturated two-layer networks can be generated by using the following restricted grammar.

$$
\begin{aligned}
\mathsf{Word} &::= \mathsf{Head} \mid \mathsf{Stick} \mid \mathsf{Cycle} & \mathsf{Stick} &::= \mathtt{12} \mid \mathsf{eStick} \mid \mathsf{oStick} \\
\mathsf{Head} &::= \mathtt{0} \mid \mathsf{eHead} \mid \mathsf{oHead} & \mathsf{eStick} &::= \mathtt{12}(\mathtt{12}+\mathtt{21})^{+}\mathtt{21} \\
\mathsf{eHead} &::= \mathtt{0}(\mathtt{12}+\mathtt{21})^{*}\mathtt{12} & \mathsf{oStick} &::= \mathtt{21}(\mathtt{12}+\mathtt{21})^{+}\mathtt{12} \\
\mathsf{oHead} &::= \mathtt{0}(\mathtt{12}+\mathtt{21})^{*}\mathtt{21} & \mathsf{Cycle} &::= \mathtt{12}(\mathtt{12}+\mathtt{21})^{+}
\end{aligned}
\tag{2}
$$

Furthermore, sentences are multi-sets M such that:

- if M contains the words $\mathtt{0_h}$ or $\mathtt{12_s}$, then all other elements of M are cycles;
- if M contains an eHead or eStick, then it contains no oHead or oStick.

With these restrictions, generating all saturated networks for $n \leq 20$ can be done almost instantaneously. The numbers $|S_n|$ of saturated two-layer networks and $|R(S_n)|$ of equivalence classes modulo permutation are given in rows two and four of Table 3.

n	3	4	5	6	7	8	9	10	11	12	13	14
$\|G_n\|$	4	10	26	76	232	764	2,620	9,496	35,696	140,152	568,504	2,390,480
$\|S_n\|$	2	4	10	28	70	230	676	2,456	7,916	31,374	109,856	467,716
$\|R(G_n)\|$	4	8	16	20	52	61	165	152	482	414	1,378	1,024
$\|R(S_n)\|$	2	2	6	6	14	15	37	27	88	70	212	136
$\|R_n\|$	1	2	4	5	8	12	22	21	48	50	117	94

n	15	16	17	18	19
$\|G_n\|$	10,349,536	46,206,736	211,799,312	997,313,824	4,809,701,440
$\|S_n\|$	1,759,422	7,968,204	31,922,840	152,664,200	646,888,154
$\|R(G_n)\|$	3,780	2,627	10,187	6,422	26,796
$\|R(S_n)\|$	494	323	1,149	651	2,632
$\|R_n\|$	262	211	609	411	1,367

Table 3: Values of $|G_n|$, $|R(G_n)|$, $|S_n|$, $|R(S_n)|$ and $|R_n|$ for $n \leq 19$.

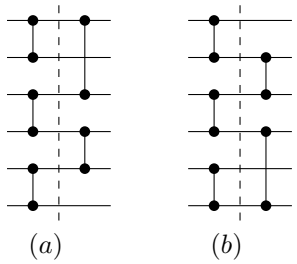

Figure 8: A two-layer, 6-channel comparator network (a) and its reflection (b).

4.3 Reflections

In the previous section, we have shown that it suffices to consider only representatives of saturated networks, where two (saturated) networks are equivalent if their corresponding graphs are isomorphic. In this section, we extend the notion of equivalence by noting that a (vertical) reflection of a sorting network is also a sorting network.

Formally, the reflection of a comparator network C on n channels is the network C^R obtained from C by replacing each comparator (i, j) with $(n - j + 1, n - i + 1)$. Note that this operation preserves the size and the depth of C. Figure 8 shows a two-layer, 6-channel comparator network and its reflection. These networks are both elements of $R(S_6)$, corresponding to the two different words $\mathtt{211212}_\mathtt{s}$ and $\mathtt{121221}_\mathtt{s}$, and as such are not equivalent using the theory developed so far.

Given a vector $\bar{x} \in \mathcal{B}^n$, denote by $\overline{\bar{x}^R}$ the vector obtained from $\bar{x}$ by reversing and complementing each bit. For example, $\overline{100^R} = 110$.

Lemma 8. *Let C be a comparator network on n channels and C^R be its reflection. Then $\bar{x} \in \mathsf{outputs}(C)$ if and only if $\overline{\bar{x}^R} \in \mathsf{outputs}(C^R)$.*

Proof. By induction on the size of C. For the empty network the result is trivial. Assume the result holds for C and consider the network $C; (i, j)$. Let $\bar{x} \in \mathcal{B}^n$ and $\bar{y} = C(\bar{x})$; the induction hypothesis guarantees that $\overline{\bar{y}^R} \in \mathsf{outputs}(C^R)$. Then, the comparator (i, j) will change $\bar{y}$ if and only if the comparator $(n-j+1, n-i+1)$ changes $\overline{\bar{y}^R}$, interchanging the corresponding positions in both sequences. This establishes the thesis for $C; (i, j)$. □

It follows from Lemma 8 that C can be extended to a sorting network of some depth d if and only if C^R can be extended to a sorting network of depth d. Thus, we can further reduce the number of candidate two-layer prefixes by eliminating those that are reflections of others.

Corollary 4. *Let S'_n be any subset of saturated two-layer networks on n channels containing only one of C and C^R for each $C \in S_n$. Then both S'_n and $R(S'_n)$ form complete set of filters.*

Yet again, particular sets S'_n and $R(S'_n)$ can be constructed syntactically by considering the word representation. Let C be a saturated network such that $\mathsf{word}(C)$ contains at least one Head or Stick word. Reflection transforms min-channels into max-channels and vice-versa, so the reflection of an oHead (respectively oStick) is an eHead (resp. eStick), and conversely. We can thus restrict the grammar defining saturated two-layer networks (2) to the following, which allows neither eHeads nor eSticks.

$$\begin{aligned} \mathsf{Word} &::= \mathtt{0} \mid \mathsf{oHead} \mid \mathtt{12} \mid \mathsf{oStick} \mid \mathsf{Cycle} \\ \mathsf{oStick} &::= \mathtt{21}(\mathtt{12}+\mathtt{21})^{+}\mathtt{12} \\ \mathsf{oHead} &::= \mathtt{0}(\mathtt{12}+\mathtt{21})^{*}\mathtt{21} \\ \mathsf{Cycle} &::= \mathtt{12}(\mathtt{12}+\mathtt{21})^{+} \end{aligned} \tag{3}$$

This handles the case of Head- and Stick-words. It remains to consider the reflections of Cycle-words. Since reflection transforms min-channels into max-channels and conversely, the word corresponding to the reflection of a cycle can be obtained by interchanging 1s and 2s in the word corresponding to the cycle, and then shifting and possibly reversing the result to obtain the lexicographically smallest representative of that cycle. As it turns out, this will typically yield the original word; in particular, for $n < 12$, this is always the case, as the lemma below states.

Given a word w, denote by $\overline{w}$ the word obtained by interchanging 1s and 2s in w, and by w^R the reverse word to w.

Lemma 9. *Let C be a network on an even number n of channels consisting of a connected cycle. If $n < 12$, then C is equivalent to C^R.*

Proof. Every Cycle-word can be written as $w = \mathtt{12}(\mathtt{12})^{k_1}\mathtt{21}(\mathtt{12})^{k_2}\mathtt{21}\ldots\mathtt{21}(\mathtt{12})^{k_n}$. Then $\overline{w}^R$ is the word $(\mathtt{12})^{k_n}\mathtt{21}\ldots\mathtt{21}(\mathtt{12})^{k_2}\mathtt{21}(\mathtt{12})^{k_1}\mathtt{12}$, and it can always be shifted into w unless k_1, k_2 and k_3 are all distinct. The shortest word where k_1, k_2 and k_3 are all distinct is 122112211212, where $k_1 = 0$, $k_2 = 1$ and $k_3 = 2$, corresponding to a cycle on 12 channels. □

Call a word w *asymmetric* if $\mathsf{word}(\mathsf{net}(w)^R) \neq w$. The previous result states that the shortest asymmetric Cycle-word has length 11. Table 4 indicates the number A_n of asymmetric cycles on n channels, modulo reflection. These values are computed by generating all Cycle-words of length n using the grammar in Equation (3) and testing whether they are asymmetric.

This sequence has been described previously in [12], which describes the computation of the number of possible crystal structures with particular kinds of symmetries. Sequence A_n above corresponds precisely to the cardinality of the space group $P6_3/mc$, which is computed by a symbolic representation whose specification exactly matches that of A_n. The values given in [12] however differ from ours for the values $36 + 6k$, for integers $k \geq 0$. We believe that this is due to errors in the original computations described in [12].

n	12	14	16	18	20	22	24	26	28	30	32	34	36	38	40
A_n	1	1	4	7	18	31	70	126	261	484	960	1,800	3,515	6,643	12,852

Table 4: Number of asymmetric Cycle-words on n channels, modulo reflection.

Definition 8. *The set of two-layer representative prefixes R_n is the set of all networks generated from sentences s such that:*

- *all words $w \in s$ are generated from the grammar in Equation* (3)*;*
- *all Stick-words $w \in s$ satisfy $w < w^R$;*

- *all Cycle-words $w \in s$ satisfy $w = \mathsf{word}(\mathsf{net}(w))$;*
- *if s contains $\mathtt{0_h}$ or $\mathtt{12_s}$, then all other words in s are Cycle-words;*
- *if s does not contain* oHead *or* oStick *words and k is the shortest length for which s contains only one asymmetric Cycle-word w of length k (possibly with high multiplicity), then $w < \mathsf{word}(\mathsf{net}(w)^R)$.*

Theorem 7. *The set R_n contains a representative for all equivalence classes of networks on n channels up to reflection. Furthermore, if R_n contains two networks that are equivalent up to reflection, then those networks are represented by a sentence s such that:*

- *s contains only Cycle-words;*
- *for every length k, the number of distinct asymmetric words of length k in s is not 1.*

Proof. By construction, R_n contains one network from each equivalence class in G_n modulo reflection. Furthermore, since oSticks and oHeads are reflected to eSticks and eHeads, no network containing any of these words can be represented together with its reflection. Likewise, if a network containing only cycles contains only one asymmetric cycle of some length, then the last criterium will guarantee that either itself or its reflection will not be included in R_n. □

Corollary 5. *For every n, the set R_n is a complete sets of filters on n channels.*

It is possible to find two asymmetric cycles A and B of the same length such that $\mathsf{word}(A) < \mathsf{word}(B)$ and $\mathsf{word}(B^R) < \mathsf{word}(A^R)$, hence for some n the set R_n still may contain redundancy. However, any network with two such cycles has at least 32 channels, since each such cycle requires at least 16 channels. In particular, for $n \leq 31$ the sets R_n contain no redundancy.

The last line of Table 3 shows the number of representatives, $|R_n|$, one needs to analyze to solve the optimal-depth problem for sorting networks of size up to 19. We can compute the sets R_n efficiently (in under two hours) for $n \leq 40$. For greater n, the execution time begins to grow due to the complexity of the test for asymmetric cycles. This could be reduced by techniques such as tabling; however, the interest of such optimizations is limited, since 40 is well beyond the scope of current techniques for solving the optimal-depth problem.

The sets of comparator networks, R_n, for $n \leq 16$ can be downloaded from `http://www.cs.bgu.ac.il/~mcodish/Papers/Appendices/SortingNetworks/`.

5 Propositional Encoding of Sorting Networks

In the previous section, we showed how to compute a set of two-layer networks, R_n, that is complete when looking for optimal-depth sorting networks on n channels. In this section, we employ this result to represent existence of sorting networks of a given depth by propositional formulas. Using a SAT solver on the obtained formulas, we can both find the optimal-depth sorting networks for $n \leq 16$ channels and prove their optimality.

For experimentation we used a cluster of Intel E8400 cores clocked at 2 GHz each. Each of the cores in the cluster has computational power comparable to a core on a standard desktop computer. Although we used all of the cores of the cluster in our experimentation, each individual instance was run on a single core. All of the times indicated in all of our results, detailed in the following, are obtained using a single core of the cluster.

Morgenstern et al. [16] observed that an n-channel comparator network of a given depth d can be represented by a propositional formula such that the existence of a depth-d, n-channel, sorting network is equivalent to that formula's satisfiability. We improve upon the work in [16], and give a more natural translation to propositional formulas. In contrast to the encoding proposed in [16], which did not prove sufficient to solve any open instances beyond $n = 10$, ours facilitates the search for optimal-depth sorting networks with up to 16 channels.

The encoding uses (similarly to the one in [16]) the zero-one principle (Lemma 1), which states that, when checking whether a comparator network is a sorting network, it suffices to consider only its outputs on Boolean inputs. We can represent the effect of a comparator on Boolean values $x, y \in \mathcal{B}$ as $\min(x, y) = x \wedge y$ and $\max(x, y) = x \vee y$.

5.1 Optimal-depth sorting networks

We now describe the construction of a propositional formula $\Psi(n, d)$ that is satisfiable if and only if an n-channel sorting network of depth d exists. Moreover, the formula $\Psi(n, d)$ has the property that: if the formula is satisfiable, then an n-channel sorting network of depth d can be easily extracted from a satisfying assignment.

The formula uses the following set of Boolean variables, specifying the position of the comparators in the network:

$$\mathtt{V}_n^d = \left\{ \; c_{i,j}^\ell \;\middle|\; 1 \le \ell \le d, 1 \le i < j \le n \; \right\}$$

where the intention is that $c_{i,j}^\ell$ is true if and only if the network contains a comparator between channels i and j at depth ℓ.

Further, to facilitate the specification of the encoding, we introduce an additional set of Boolean variables capturing which channels are "used" at a given layer:

$$\mathtt{U}_n^d = \left\{ \; u_k^\ell \;\middle|\; 1 \le \ell \le d, 1 \le k \le n \; \right\}$$

where the intention is that u_k^ℓ is true if and only if there is some comparator on channel k at level ℓ.

The following formula enforces the relation between the variables in $\mathtt{U}_n^d$ and in $\mathtt{V}_n^d$:

$$\varphi_{n,d}^{used} \quad = \bigwedge_{\substack{1 \le \ell \le d \\ 1 \le k \le n}} u_k^\ell \leftrightarrow \bigvee \mathit{incident}_k^\ell(\mathtt{V}_n^d)$$

where $\mathit{incident}_k^\ell(\mathtt{V}_n^d)$ denotes, for each channel k and level ℓ, which variables in $\mathtt{V}_n^d$ correspond to comparators incident to channel k at layer ℓ:

$$\mathit{incident}_k^\ell(\mathtt{V}_n^d) = \left\{ \; c_{i,j}^\ell \;\middle|\; c_{i,j}^\ell \in \mathtt{V}_n^d, \; i = k \text{ or } j = k \; \right\}$$

Now, a representation of a comparator network is valid if at each layer every channel is used by at most one comparator. This is enforced by the following formula:

$$\varphi_{n,d}^{valid} \quad = \bigwedge_{\substack{1 \le \ell \le d \\ 1 \le k \le n}} \mathit{at\_most\_one}(\mathit{incident}_k^\ell(\mathtt{V}_n^d))$$

where for any set of Boolean variables $B = \{b_1, \ldots, b_n\}$, the formula $\mathit{at\_most\_one}(B)$ signifies that at most one of the variables in B takes the value *true*. We adopt the straightforward encoding:

$$\mathit{at\_most\_one}(\{b_1, \ldots, b_n\}) \quad = \bigwedge_{i<j} (\neg b_i \vee \neg b_j)$$

Given sets of Boolean variables $\bar{x} = \{x_1, \ldots, x_n\}$ and $\bar{y} = \{y_1, \ldots, y_n\}$, we use the following formula to express that $\{y_1, \ldots, y_n\}$ is obtained from $\{x_1, \ldots, x_n\}$ by applying the ℓ-th layer of the network:

$$\varphi_{n,d}^\ell(\bar{x}, \bar{y}) = \underbrace{\bigwedge_{i<j} c_{i,j}^\ell \rightarrow \left(\bigwedge \begin{array}{l} y_i \leftrightarrow x_i \wedge x_j \\ y_j \leftrightarrow x_i \vee x_j \end{array} \right)}_{a} \quad \wedge \quad \underbrace{\bigwedge_k \neg u_k^\ell \rightarrow (x_k \leftrightarrow y_k)}_{b}$$

The left part (a) specifies that the outputs of a comparator on channels i and j are the minimum and maximum of its inputs; the right part (b) specifies that, if a channel is not incident to any comparator, then its output is equal to its input.

To express that the network sorts the input $\bar{b} = \{b_1, \ldots, b_n\}$, we introduce Boolean variables $\bar{x}^i = \{x_1^i, \ldots, x_n^i\}$ for $0 \leq i \leq d$, where $\bar{x}^i$ shall denote the values on the n channels after the ith layer of the network. We set $\bar{x}^0 = \bar{b}$, denote by $\bar{b}'$ the result of sorting the given vector $\bar{b}$, and write:

$$\varphi_{n,d}^{sort}(\bar{b}) = \bigwedge_{\ell=1}^{d} \varphi_{n,d}^{\ell}(\bar{x}^{i-1}, \bar{x}^i) \wedge \bigwedge_{i=1}^{n} \bar{x}_i^d \leftrightarrow \bar{b}'_i$$

Then, given a set of Boolean inputs $X \subseteq \mathcal{B}^n$, the following formula is satisfiable if and only if there is a depth-d, n-channel network sorting all inputs from X:

$$\Psi(n, d, X) = \varphi_{n,d}^{used} \wedge \varphi_{n,d}^{valid} \wedge \bigwedge_{\bar{b} \in X} \varphi_{n,d}^{sort}(\bar{b}) \tag{4}$$

A sorting network must sort all Boolean inputs. Hence, the following result holds.

Lemma 10. *There exists a sorting network with n channels and depth d if and only if the formula $\Psi(n, d, \mathcal{B}^n)$ is satisfiable.*

Later we show how to modify an encoding based on the formula in Equation (4) to make use of the results from Section 4. First, to improve the performance of an actual SAT solver on such an encoding, we introduce several additional optimizations which we now briefly describe.

No redundant comparators. If a comparator occurs in two consecutive layers of a network, then the second one has no effect. To prevent the placement of redundant comparators, we introduce the following symmetry breaking formula:

$$\sigma_1 = \bigwedge_{\substack{1 \leq \ell < d \\ 1 \leq i < j \leq n}} \neg c_{i,j}^{\ell} \vee \neg c_{i,j}^{\ell+1}$$

Eager comparator placement. If a comparator is positioned on a pair of channels at level ℓ that are not used at level $\ell - 1$, then it can be "slided to the previous layer". To prevent the placement of such sliding comparators, we introduce the following symmetry breaking formula:

$$\sigma_2 = \bigwedge_{\substack{1 < \ell \leq d \\ 1 \leq i < j \leq n}} c_{i,j}^{\ell} \rightarrow u_i^{\ell-1} \vee u_j^{\ell-1}$$

All adjacent comparators. Exercise 5.3.4.35 in [9] states that all comparators of the form $(i, i+1)$ must be present in a sorting network. To this end, we add the following (redundant) formula:

$$\sigma_3 = \bigwedge_{1 \leq i < n} \left(c_{i,i+1}^1 \vee c_{i,i+1}^2 \vee \cdots \vee c_{i,i+1}^d\right)$$

Only unsorted inputs. Let $\mathcal{B}_{un}^n$ denote the subset of $\mathcal{B}^n$ consisting of unsorted sequences. Then it is possible to refine the formula in Lemma 10 by replacing the set of all Boolean inputs $\mathcal{B}^n$ with the set of unsorted inputs $\mathcal{B}_{un}^n$. This is the case as sorted sequences are unchanged regardless of the positioning of the comparators. Observe that $|\mathcal{B}_{un}^n| = 2^n - n - 1$, and as noted by Chung and Ravikumar [5], this is the size of the smallest test set possible needed to determine whether a comparator network is a sorting network.

Optimized CNF generation. Our encodings are generated using the BEE finite-domain constraint compiler. BEE is described in several recent papers [13, 14, 15]. BEE facilitates solving finite-domain constraints by encoding them to CNF and applying an underlying SAT solver.

In BEE constraints are modeled as Boolean functions which propagate information about equalities between Boolean literals. This information is then applied to simplify the CNF encoding of the constraints. BEE is written in Prolog, and applies (in our configuration) the underlying SAT solver CryptoMiniSAT [19].

Experiment I

In our first experiment, we used the encoding of Equation (4). Here, the formula $\Psi(n, d, X)$, together with the above described optimizations, is instantiated for various values of n, d and X. Table 5 presents the results, where each instance (2 per line in the table) is run on a single thread of the cluster. For each n (number of channels) with $5 \leq n \leq 12$, there is a row in the table that indicates: the depth of the network we seek (d and d'), the encoding time (using BEE), the size of the resulting CNF (number of clauses and variables), and the SAT-solving time. Times are indicated in seconds. The left side of the table details satisfiable instances where we seek a sorting network of optimal depth d. For $1 \leq n \leq 10$, we take for d the known optimal depth, and for $n > 10$ we take the best known upper bound (see Table 1). The right side of the table details the (suspected to be) unsatisfiable instances, where d' is one less than the value d.

Observe that, when $n = 12$, in the search for a sorting network of depth 8, the encoding creates a CNF with circa 8.9 million clauses, and a solution is found after about 3.5 hours. On the other hand, for $n = 11$ the SAT solver is not able to prove unsatisfiability of $\Psi(11, 7, \mathcal{B}^{11}_{un})$ even after one week of computation, and similarly for $\Psi(12, 7, \mathcal{B}^{12}_{un})$. So this encoding suffices to prove depth optimality of networks on up to 10 channels. We now show how to extend it to handle more channels.

5.2 Optimal-depth sorting networks given a prefix

Recall that, to show the existence of a sorting network of a given size, it suffices to restrict attention to networks with a fixed first layer (Lemma 2). Furthermore (Corollary 5), it suffices to focus on second layers from the set R_n.

We now show how to capitalize on these results. In general, let n be the number of channels and d be the depth of a particular comparator network. Then, given a prefix C consisting of layers $C = L_1; L_2; \cdots ; L_{d'}$ we can encode the property that the prefix of the network is C by the following formula:

$$\varphi^{\mathit{fixed}}(n, d, C) = \bigwedge_{\substack{1 \leq \ell < d' \\ 1 \leq i < j \leq n}} c^{\ell}_{i,j} \leftrightarrow (i, j) \in L_{\ell} \tag{5}$$

where the conjuncts fix the values of the Boolean variables $c^{\ell}_{i,j}$ in $\mathtt{V}^d_n$ to correspond to the positions of the comparators in the given prefix C.

Given a set of Boolean inputs $X \subseteq \mathcal{B}^n$, there is an n-channel, depth-d network with prefix C sorting all inputs from X if and only if the following formula is satisfiable.

$$\Psi_C(n, d, X) = \Psi(n, d, X) \wedge \varphi^{\mathit{fixed}}(n, d, C) \tag{6}$$

	optimal sorting networks (sat)					smaller networks (unsat)				
n	d	BEE	#clauses	#vars	SAT	d'	BEE	#clauses	#vars	SAT
5	5	0.09	4965	761	0.01	4	0.08	3702	550	0.01
6	5	0.31	15353	1911	0.04	4	0.23	11417	1374	0.03
7	6	1.14	55758	5946	0.14	5	0.83	44330	4634	0.97
8	6	3.55	153125	14058	1.35	5	2.47	121639	10946	1.83
9	7	10.06	487489	39761	9.51	6	8.56	404176	32544	629.04
10	7	25.17	1247335	90589	93.40	6	22.02	1033821	74136	925.30
11	8	85.42	3643870	240258	518.61	7	64.59	3110693	203313	∞
12	8	234.39	8899673	533226	12343.21	7	185.27	7596239	451212	∞

Table 5: SAT-solving for n-channel, depth-d sorting networks: each instance runs on a single thread, BEE compile times and SAT-solving times are in seconds (timeout is 1 week of computation).

Note that Boolean sequences sorted after the application of C remain sorted regardless of the positioning of the comparators in the subsequent layers. Thus, to show that there is a sorting network on n channels with depth d that begins with prefix C, it suffices to restrict the set X to Boolean sequences that are unsorted after application of the prefix C. Letting $\mathcal{B}^n_{un(C)} \subseteq \mathcal{B}^n$ denote the set of such sequences, the following result holds.

Lemma 11. *There exists a sorting network on n channels with depth d and prefix C if and only if the formula $\Psi_C\left(n, d, \mathcal{B}^n_{un(C)}\right)$ is satisfiable.*

Experiment II

According to Lemma 2 we can fix the first level of the network, and thus apply the encoding $\Psi_C\left(n, d, \mathcal{B}^n_{un(C)}\right)$ where C consists of a single maximal layer on n channels. We take C to be $F'_n = \left\{ \; (i, n-i+1) \,\middle|\, 1 \leq i \leq \left\lfloor \frac{n}{2} \right\rfloor \; \right\}$. Table 6 illustrates the results for the appropriate instances of the formula $\Psi_C\left(n, d, \mathcal{B}^n_{un(F'_n)}\right)$. Each instance (2 per line in the table) is run on a single thread of the cluster. As in Table 5, the satisfiable instances are described on the left and the unsatisfiable instances on the right.

	optimal sorting networks (sat)					smaller networks (unsat)				
n	d	BEE	#clauses	#vars	SAT	d'	BEE	#clauses	#vars	SAT
5	5	0.04	1366	238	0.00	4	0.01	915	151	0.00
6	5	0.09	3137	441	0.01	4	0.04	2083	276	0.01
7	6	0.48	12699	1509	0.06	5	0.15	9487	1089	0.03
8	6	0.97	26414	2682	0.17	5	0.35	19680	1930	0.06
9	7	3.34	90846	8150	1.25	6	1.41	72337	6353	1.00
10	7	6.23	177067	14091	17.15	6	2.81	140847	10978	1.83
11	8	18.11	547708	39386	104.16	7	9.89	454563	32245	282.04
12	8	38.84	1018902	66206	211.51	7	17.02	845232	54192	521.62
13	9	112.81	2927622	174766	1669.88	8	49.26	2500930	147902	∞
14	9	110.95	5264817	288609	56654.37	8	90.63	4496413	244234	∞

Table 6: SAT-solving for n-channel, depth-d sorting networks with 1-layer filters: each instance runs on a single thread, BEE compile times and SAT-solving times are in seconds (timeout is 1 week of computation).

Note that the CNFs for this experiment are smaller than in the previous experiment as fixing the first layer of the network reduces the set of unsorted inputs considered to $\mathcal{B}^n_{un(F'_n)}$. This derives from the fact that, as discussed before, sorted inputs (here, to the second layer) can be ignored in the encoding. For example, for $n = 12$ we reduce the CNF sizes from 8.9 and 7.6 million (see Table 5) to 1.02 and 0.8 million clauses, respectively.

Observe that the encoding based on one layer filters suffices to prove depth optimalily of networks for $n = 11$ and $n = 12$ channels in under 10 minutes with computational power equivalent to that of a standard desktop computer.

Experiment III

In our third experiment, we capitalize on Corollary 5, which states that R_n is a complete set of two layer filters on n channels, i.e. that there exists an n channel sorting network of depth d if and only if there exists one that extends one of the prefixes $C \in R_n$.

As an example, for $n = 13$, $|R_{13}| = 117$, and so, to determine whether there exists a 13-channel, depth-8 sorting network, it suffices to determine whether any one of 117 independent SAT instances $\Psi_C\left(n, d, \mathcal{B}^n_{un(C)}\right)$, for $C \in R_{13}$, is satisfiable.

In Table 7 we illustrate results for the instances with optimal depth d for $1 \leq n \leq 10$, and the best known upper bound d for $11 \leq n \leq 16$. We consider the two-layer filters in the sets R_n as described in Section 4. For the row corresponding to n channels, we have $|R_n|$ instances, and each instance is run on a single thread from the cluster. The left part of the table describes the fastest satisfiable instance from the $|R_n|$ instances. The instance number

given in the third column indicates a particular filter $C \in R_n$ (the instances are detailed at `http://www.cs.bgu.ac.il/~mcodish/Papers/Appendices/SortingNetworks/twoLayerFilters.pl`). When running the instances with the full capacity of the cluster, we can abort the computation as soon as the first satisfiable instance is found. The right part of the table specifies the total cost of the computation, and indicates the time required for each value of n on a single core. Here, we indicate the total compile times and SAT-solving times for all $|R_n|$ instances. Each instance was limited to run for 24 hours on a single core, and $\infty(k)$ indicates that k instances terminated within 24 hours (each on a single core).

For the unsatisfiable instances, we introduce one additional optimization. Consider again Equation (6). A sorting network with prefix C must sort all of its unsorted inputs $\mathcal{B}^n_{un(C)}$. However, if we consider any specific subset of $B \subseteq \mathcal{B}^n_{un(C)}$ and show that there is no comparator network that sorts the elements of B, then there is also no comparator network that sorts all the unsorted inputs.

In particular, we consider length-n Boolean sequences that have sufficiently long prefixes of zeroes and suffixes of ones. Given an integer $w < n$ and a set $B \subseteq \mathcal{B}^n$, we denote the set $B{\restriction}w = \left\{ b \in B \,\middle|\, b = 0^{\ell_1}.\mathcal{B}^{n-w}.1^{\ell_2}, \ell_1 + \ell_2 = w \right\}$, which we refer to as the *windows* of size w of B.

Table 8 depicts results for the instances with the set of inputs equal to $\mathcal{B}^n_{un(C)}{\restriction}w$ and depth $d' = d - 1$ where d is the known optimal depth for $1 \leq n \leq 10$, and the best known upper bound d for $11 \leq n \leq 16$. We consider the two-layer filters in the sets R_n as described in Section 4. For the row corresponding to n channels, we again have $|R_n|$ instances, and each instance is run on a single thread from the cluster. The left part of the table describes the slowest unsatisfiable instance from the $|R_n|$ instances, including the largest window size w for which unsatisfiability is obtained. The table also specifies, in the fourth column, the index of the slowest instance. For the unsatisfiable instances, we need to run all instances to determine that all are unsatisfiable, and the parallel cost is the time of the slowest instance (in the left part of the table). The right part of the table specifies the total cost of the computation, indicating the time required for each value of n on a single core. Here we indicate the total compile times and SAT-solving times for all $|R_n|$ instances.

While we used multiple threads on a cluster for our experiments, the two instances relevant for our results, $n = 11$ and $n = 13$, could be run on a single thread on a desktop computer in 2 minutes and 10 hours, respectively. Once we show that $T(11) = 8$, it follows from the known bounds (Table 1) that also $T(12) = 8$, because T is monotonic. Likewise, once we show that $T(13) = 9$, it follows that also $T(14) = T(15) = T(16) = 9$. The motivation for also computing them was to show that this approach actually scales up to $n = 16$.

			optimal depth: fastest satisfiable instance					total solving times	
n	$\lvert R_n\rvert$	d	ins.	BEE	#clauses	#vars	SAT	BEE	SAT
5	4	5	3	0.01	534	97	0.00	0.03	0.01
6	5	5	5	0.01	1047	156	0.00	0.04	0.01
7	8	6	3	0.06	4537	569	0.01	0.49	0.08
8	12	6	10	0.07	6952	740	0.02	1.01	0.18
9	22	7	5	0.38	26019	2447	0.06	10.52	4.66
10	21	7	19	0.61	50573	4216	0.36	2.28	2.02
11	48	8	10	2.67	171357	13129	0.60	126.99	753.71
12	50	8	43	2.77	206776	14088	4.08	57.07	481.13
13	117	9	112	13.28	922363	56679	10.71	1711.99	38185.55
14	94	9	86	37.80	1124987	64318	123.13	6206.11	$\infty(43)$
15	262	9	169	84.90	2684977	139181	19737.38	33176.39	$\infty(1)$
16	211	9	188	116.36	3179978	155456	30509.58	46968.71	$\infty(1)$

Table 7: SAT-solving for n-channel, depth-d sorting networks with 2-layer filters. These are the satisfiable instances (at least for one $C \in R_n$): fastest satisfiable instances detailed on the left; and total costs on the right: BEE compile times and SAT-solving times are in seconds. Here, $\infty(k)$ indicates that k instances terminated within 24 hours (each on a single core).

			d' < optimal depth: slowest (unsat) instance						total solving times	
n	$\lvert R_n \rvert$	d'	ins.	w	BEE	#clauses	#vars	SAT	BEE	SAT
5	4	4	1	2	0.00	268	44	0.00	0.02	0.00
6	5	4	1	2	0.01	511	67	0.00	0.02	1.00
7	8	5	6	2	0.04	2965	348	0.01	0.30	1.04
8	12	5	4	3	0.09	4423	458	0.01	0.80	4.09
9	22	6	16	3	0.20	14716	1416	0.05	4.16	0.75
10	21	6	1	4	0.41	20027	1815	0.09	6.35	5.08
11	48	7	46	4	0.73	52365	4314	1.41	53.79	52.26
12	50	7	21	5	1.02	62051	4826	2.39	93.79	72.56
13	117	8	77	3	14.68	464035	29958	749.27	1047.93	35726.26
14	94	8	16	4	20.27	448903	27473	2627.60	2342.96	81533.49
15	262	8	189	7	8.32	278312	18217	746.42	3491.06	127062.36
16	211	8	112	7	22.85	453810	27007	1756.29	4448.66	152434.55

Table 8: SAT-solving for n-channel, depth-d' sorting networks with 2-layer filters. These are the unsatisfiable instances (for all $C \in R_n$): slowest instances detailed on the left; and total costs on the right: BEE compile times and SAT-solving times are in seconds; and w is window size.

6 Conclusions and future work

We have shown that $T(11) = T(12) = 8$ and $T(13) = T(14) = T(15) = T(16) = 9$, i.e., we have proven that the previously known upper bounds on the optimal depth of n-channel sorting networks are tight for $11 \leq n \leq 16$. This closes the six smallest open instances of the optimal-depth sorting network problem, thereby proving depth optimality of the sorting networks for $n \leq 16$ given in [9] more than four decades ago.

The next smallest open instance of the optimal-depth sorting network problem is for $n = 17$ where the best known upper bound is 11. Attempting to show that there is no sorting network of depth 10 requires analyzing the SAT encodings given the networks in R_{17}. The resulting 609 formulas have more than five million clauses each, and none could be solved within a couple of weeks. It appears that establishing the optimal depth of sorting networks with more than 16 channels is a hard challenge that will require prefixes with more than 2 layers.

The encoding into SAT that we propose in this paper is of size exponential in the number of channels, n. This is also the case for the encoding presented in [16]. The encoding is of the form $\exists\forall\varphi$ (does there *exist* a network that sorts *all* of its inputs), and is easily shown to be in Σ_2^P. We expect that, similar to the problem of circuit minimization [20], it is also complete in Σ_2^P, although we have not succeeded to prove this. We do not expect that there exists a polynomial size encoding to SAT.

Acknowledgements

We thank Donald E. Knuth for suggesting the idea of using reflections to reduce the number of representative two-layer networks.

References

[1] M. Ajtai, J. Komlós, and E. Szemerédi. An O(n log n) sorting network. In *Proceedings of the fifteenth annual ACM symposium on Theory of Computing*, STOC '83, pages 1–9, 1983.

[2] K. E. Batcher. Sorting networks and their applications. In *Proceedings of the spring joint computer conference*, AFIPS '68 (Spring), pages 307–314. ACM, 1968.

[3] Daniel Bundala and Jakub Závodný. Optimal sorting networks. In *LATA 2014*, LNCS 8370, pages 236–247. Springer, 2014.

[4] Sung-Soon Choi and Byung Ro Moon. Isomorphism, normalization, and a genetic algorithm for sorting network optimization. In *GECCO 2002*, pages 327–334. Morgan Kaufmann, 2002.

[5] Moon Jung Chung and B. Ravikumar. Bounds on the size of test sets for sorting and related networks. *Discrete Mathematics*, 81(1):1–9, 1990.

[6] Michael Codish, Luís Cruz-Filipe, Michael Frank, and Peter Schneider-Kamp. Twenty-five comparators is optimal when sorting nine inputs (and twenty-nine for ten). *CoRR*, abs/1405.5754, 2014.

[7] Michael Codish, Luís Cruz-Filipe, and Peter Schneider-Kamp. The quest for optimal sorting networks: Efficient generation of two-layer prefixes. In *SYNASC*, 2014. To appear.

[8] Robert W. Floyd and Donald E. Knuth. The Bose–Nelson sorting problem. In *A survey of combinatorial theory*, pages 163–172. North-Holland, 1973.

[9] Donald E. Knuth. *The Art of Computer Programming, Volume III: Sorting and Searching*. Addison-Wesley, 1973.

[10] Eugene M. Luks. Isomorphism of graphs of bounded valence can be tested in polynomial time. *Journal of Computer and System Sciences*, 25(1):42–65, 1982.

[11] Brendan D. McKay and Adolfo Piperno. Practical graph isomorphism, II. *Journal of Symbolic Computation*, 60:94–112, 2014.

[12] T.J. McLarnan. The numbers of polytypes in close-packings and related structures. *Zeitschrift für Kristallographie*, 155(3–4):269–291, January 1981.

[13] Amit Metodi and Michael Codish. Compiling finite domain constraints to SAT with BEE. *TPLP*, 12(4-5):465–483, 2012.

[14] Amit Metodi, Michael Codish, Vitaly Lagoon, and Peter J. Stuckey. Boolean equi-propagation for optimized SAT encoding. In Jimmy Ho-Man Lee, editor, *CP*, volume 6876 of *LNCS*, pages 621–636, , 2011. Springer.

[15] Amit Metodi, Michael Codish, and Peter J. Stuckey. Boolean equi-propagation for concise and efficient SAT encodings of combinatorial problems. *J. Artif. Intell. Res. (JAIR)*, 46:303–341, 2013.

[16] Andreas Morgenstern and Klaus Schneider. Synthesis of parallel sorting networks using SAT solvers. In *MBMV 2011*, pages 71–80. OFFIS-Institut für Informatik, 2011.

[17] I. Parberry. A computer assisted optimal depth lower bound for sorting networks with nine inputs. In *Proceedings of the 1989 ACM/IEEE Conference on Supercomputing*, Supercomputing '89, pages 152–161, New York, NY, USA, 1989. ACM.

[18] Ian Parberry. A computer-assisted optimal depth lower bound for nine-input sorting networks. *Mathematical Systems Theory*, 24(2):101–116, 1991.

[19] Mate Soos. CryptoMiniSAT, v2.5.1. `http://www.msoos.org/cryptominisat2`, 2010.

[20] Christopher Umans. The minimum equivalent DNF problem and shortest implicants. In *FOCS*, pages 556–563. IEEE Computer Society, 1998.